بسمه تعالی

تمرین سری دهم درس یادگیری عمیق

پوریا محمدی نسب

۴۰۰۷۲۲۱۳۸

فهرست

سوال ۱.

یکی از مشکلات رایج مکانیزم توجه، مخصوصا هنگامی که متن ورودی در طرف encoder طولانی باشد، عدم توانایی این مکانیزم در پرداختن به تکه‌های مختلف متن ورودی است. مثلا ممکن است در تمامی گام‌های decoder، مکانیزم توجه فقط به یک یا دو کلمه‌ی خاص امتیاز بسیار بالایی بدهد و فقط آنها را در نظر بگیرد. در این صورت مدل قادر نخواهد بود که از تمامی متن ورودی استفاده کند. برای حل این مشکل چه راهکاری پیشنهاد می‌دهید؟ توضیح دهید.

(راهنمایی: شما می‌توانید یک جمله‌ی جدید به تابع خطا/هزینه‌ی مدل اضافه کنید).

مشکلی که به دلیل این رخداد پیش می‌آید به over-translationمعروف است که در رفرنس [5] برای اولین بار مطرح شد. در این مقاله دلیل به وجودآمدن مشکل over-translation را شباهت بسیار بردارهای متن ذکرشده است که این بردارها با مکانیزم توجه یادگرفته میشوند. همچنین در این مقاله با معرفی راهکاری به نام GAtt سعی در بهبود قدرت تفکیک پذیری بردارهای متن دارد. حال به شرح دقیق‌تر GAtt میپردازیم. GAtt یا GRU-Gated Attention شامل دو لایه است.

۱. gating layer:

هدف این لایه بهبود ارائه منبع با توجه به حالت رمزگشای قبلی (S_{j-1}) به منظور محاسبه بازنمایی منبع مرتبط ترجمه است. رابطه ی زیر بیانگر این لایه است:

$$H_j^g = Gate(H, s_{j-1})$$

در رابطه بالا Gate(.) باید بتواند با ارتباط پیچیده بین منبع ورودی و ترجمه کار کند و به راحتی رابطه معنایی و جریان اطلاعات بین ورودی و ترجمه را کنترل کند.

به جای استفاده از مکانیزم gating کانولوشنی در این روش از کل ساختار GRU برای انجام این وظیفه استفاده میکنیم. که در این حالت روابط بهبود حالت رمزگشای قبلی (S_{j-1}) عبارت است از:

$$\begin{aligned}
\mathbf{z}_{ji} &= \sigma(W_z \mathbf{s}_{j-1} + U_z \mathbf{h}_i + b_z) \\
\mathbf{r}_{ji} &= \sigma(W_r \mathbf{s}_{j-1} + U_r \mathbf{h}_i + b_r) \\
\overline{h}_{ji} &= \tanh(W \mathbf{s}_{j-1} + U\,[\mathbf{r}_{ji} \odot \mathbf{h}_i] + b) \\
\mathbf{h}^g_{ji} &= (1 - \mathbf{z}_{ji}) \odot \mathbf{h}_i + \mathbf{z}_{ji} \odot \overline{h}_{ji}
\end{aligned}$$

که σ همان sigmoid و ⊙ ضرب عنصری است و دو گیت r_{ji} و z_{ji} معیاری از درجه معنادار بودن و ارتباط معنایی بین ورودی و ترجمه هستند.

۲. attention layer:

این لایه با لایه اصلی در مکانیزم توجه یکسان است و رابطه ی آن عبارت است از:

$$\mathbf{c}_j = \mathrm{Att}(\mathbf{H}^g_j, \mathbf{s}_{j-1})$$

به هر حال به جای پرداختن به بازنمایی منبع H، این لایه به بهبود گیت H^g وابسته است. همانطور که مشخص است H^g در طول decoding پویا است. باید به این نکته توجه کرد که Gate(.) یک RNN چند مرحله ای نیست و یک تابع ساده‌ی ترکیبی و یا یک RNN تک مرحله ای است پس در نتیجه از لحاظ محاسباتی بسیار موثر است. برای آموزش این مدل همانند قبل هدف بهینه سازی را ماکسیمم کردن مقدار log-likelihood در دیتای آموزشی میگذاریم و با استفاده از بهینه ساز گرادیان کاهشی تصادفی، پارامترهای مدل را بهبود میدهیم.

حال میتوان با استفاده کردن از حالت decoder قبلی به عنوان تاریخچه و منبع بازنمایی ورودی جدید مدلی تحت عنوان GAtt-Inv طراحی کرد که روابط آن برابر است با:

$$\mathbf{c}_j = \text{GAtt-Inv}(\mathbf{H}, \mathbf{s}_{j-1})$$

$$\mathbf{c}_j = \text{Att}(\mathbf{H}_j^{g\prime}, \mathbf{s}_{j-1}) \quad \mathbf{H}_j^{g\prime} = \text{Gate}(\mathbf{s}_{j-1}, \mathbf{H})$$

برای مقایسه مدل های پیشنهادی و اثبات بهبودی که در کارایی مدل دارند چند جدول نتیجه از مقاله آورده ایم:

System	SAER	AER
Tu et al. [2016]	64.25	50.50
RNNSearch	64.10	50.83
GAtt	**56.19**	**43.53**
GAtt-Inv	58.29	43.60

Table 4: SAER and AER scores of word alignments deduced by different neural systems. The lower the score, the better the alignment quality.

System	1-gram	2-gram	3-gram	4-gram
Reference	12.94	1.80	0.93	1.29
RNNSearch	19.12	5.26	3.27	2.97
GAtt	18.09	4.11	2.50	2.46
GAtt-Inv	16.79	3.39	1.99	1.94

Table 5: N-GRR scores of different systems on all test sets with N ranges from 1 to 4. The lower the score, the better the system deals with the over-translation problem.

سوال ۲.

توجه سخت و توجه نرم را با هم مقایسه کنید و بگویید کدام یک را میتوان با استفاده از روش پس انتشار خطا آموزش داد؟ چرا؟(برای نمونه می‌توانید از این مرجع کمک بگیرید.)

به طور کلی مکانیزم های توجه دو دسته کلی هستند. توجه نرم(soft attention) و توجه سخت(Hard attention). توجه نرم به **طور کامل مشتق پذیر** و قابل استفاده در روش پس انتشار است. همینطور مکانیزم توجه نرم به صورت قطعی عمل میکند و میتواند به یک سیستم موجود الحاق شود و گرادیان های آن در طول زمان به باقی قسمت های شبکه propagate شوند. در طرف مقابل مانیزم توجه سخت وجود دارد که تصادفی(stochastic) عمل میکند و به جای استفاده از کل حالت های مخفی به عنوان ورودی قسمت رمزگشا، سیستم از حالت مخفی y_i با احتمال s_i استفاده میکند. توجه سخت به صورت کلی مشتق پذیر نیست و نمیتوان از آن در روش آموزش پس انتشار استفاده کرد اما میتوان با روش Monte Carlo Sampling گرادیان ها را در توجه سخت **تخمین** زد. هر دو روش مزایا و معایب خاص خود را دارند اما به طور معمول به دلیل اینکه میتوان از توجه نرم به صورت مستقیم گرادیان را محاسبه کرد و نیازی نیست گرادیان را تخمین بزنیم معمولا توجه نرم بیشتر مورد استفاده قرار میگیرد. قابلیت مشتق پذیری توجه نرم باعث شده است تا به صورت وسیعی در کاربردهای بینایی ماشین استفاده شود. اما از نقاط قوت توجه سخت میتوان به این مورد اشاره کرد که این مکانیزم میتواند ویژگی های مهم را از اطلاعات ورودی انتخاب کند و در این کاربرد توجه سخت بسیار موثرتر و بهتر عمل میکند.

سوال ۳.

به سوالات زیر پاسخ دهید:(برگرفته از سوالات کتاب d2l)

الف)

فرض کنید یک معماری عمیق طراحی کرده‌ایم تا با چیدن لایه‌های self-attention پشت سر هم با کدگذاری موقعیتی، وابستگی‌های یک دنباله را مدلسازی کنیم. این معماری چه مشکلاتی می‌تواند داشته باشد؟

برای جواب به این سوال از مقاله رفرنس [7] استفاده میکنیم. همانطور که میدانیم عمق معماری رابطه ای مستقیم با توانایی شبکه در یافتن وابستگی و ارتباط بین زیرمجموعه های ورودی دارد. در حالت پایه (single self-attention layer) با توجه به رابطه خطی برای محاسبه attention score با محدودیت جهت پیداکردن وابستگی ها مواجه هستیم. در مقاله رفرنس [7] تئوری اثبات میشود که با افزایش لایه های self-attention ظرفیت شبکه به صورت نمایی افزایش میابد.

Theorem 1. *For $p \in [d_x]$, let $y_p^{i,L,d_x,H,\Theta}$ be the scalar function computing the pth entry of an output vector at position $i \in [N]$ of the depth-L self-attention network with embedding dimension d_x and H attention heads per layer, defined in eqs. (3) and (4). Let $sep(y_p^{i,L,d_x,H,\Theta})$ be its separation rank (section 3). If L, d_x obey $L < \log_3(d_x)$, then the following holds almost everywhere in the network's learned parameter space,* i.e. *for all values of the weight matrices (represented by Θ) but a set of Lebesgue measure zero:*

$$3^{L-2}(\log_3(d_x - H) + a) \leq \log_3\left(sep(y_p^{i,L,d_x,H,\Theta})\right) \leq \frac{3^L - 1}{2}\log_3(d_x + H) \qquad (6)$$

with $a = -L + [2 - \log_3 2]$. (note that $\log_3(d_x - H) + a > 0$ in this regime of $L < \log_3(d_x)$).

از طرفی دیگر در تئوری دوم این مقاله اثبات میشود هر چه عمق شبکه افزایش یابد و ظرفیت نیز زیاد شود قدرت بازنمایی داخلی محدود میشود.

Theorem 2. *For $y_p^{i,L,d_x,H,\Theta}$ as defined in theorem 1, if $L > \log_3(d_x)$, then the following holds almost everywhere in the network's learned parameter space,* i.e. *for all values of the weight matrices (represented by Θ) but a set of Lebesgue measure zero:*

$$\frac{1}{2}d_x \cdot L + b_1 + b_2 \leq \log_3\left(sep(y_p^{i,L,d_x,H,\Theta})\right) \leq 2d_x \cdot L + c_1 + c_2 \qquad (7)$$

with corrections on the order of L: $b_1 = -L\left(\frac{H}{2} + 1\right)$, $c_1 = L$, and on the order of $d_x \log_3(d_x)$: $b_2 = -d_x\left(1 + \frac{1}{2}\log_3\left(\frac{d_x - H}{2}\right)\right)$, $c_2 = -2d_x \cdot \log_3 d_x/2\sqrt{2e} + \log_3 d_x$.

به این معنی که اگر عمق را از یک حد آستانه ای بیشتر کنیم که این حد آستانه بر اساس پهنای شبکه است($L > \log(d_x)$) ضرفیت شبکه به صورت نمایی افزایش نمیابد و حتی ممکن است در مواردی یک شبکه کم عمق تر و با پهنای بیشتر بهتر عمل کند.

ب)

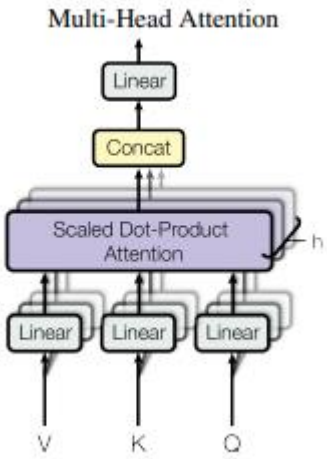
آیا می‌توان فقط با استفاده از ضرب ماتریس‌ها، یک تابع امتیازدهی جدید برای query ها و key ها با طول های برداری مختلف طراحی کرد؟

خیر- نمیشود. از آنجا که برای ها key یک value متناظر داریم پس ابعاد ماتریس key و value سطرهای یکسانی دارند. حالا اگر بردار query را در این ماتریس ضرب کنیم ابعاد متفاوتی در ماتریس key خواهیم داشت و در این ماتریس نمیتوان چنین شرطی را برقرار کرد. رفرنس [6] اثباتی است بر عدم توانایی این خواسته.

فرض کنید که به جای اینکه تابع attention را فقط یکبار اجرا کنیم با بعد d_{model} به اندازه h بار با بعد d_{model}/h بار اجرا کنیم و پس از گرفتن نتیجه حاصل این h بار اجرا را با هم concat کنیم. در رفرنس [6] این راه حل را به صورت دقیق تری توضیح داده است. حال رابطه مربوط به معماری معرفی شده مطابق زیر خواهد بود:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

و معماری معرفی شده نیز در شکل زیر آمده است و چگونگی استفاده از query، key و value ها را در h بار تکرار attention نمایش میدهد.



در شکل بالا واضح است که $d_k = d_v = d_{model}/h$.

ج)

وقتی query ها و key ها طول برداری یکسانی دارند، آیا جمع برداری می‌تواند طراحی بهتری نسبت به ضرب نقطه‌ای برای تابع امتیازدهی باشد؟ دلیل پاسخ مثبت یا منفی خود را بیان کنید.

با استفاده از ضرب در تابع امتیاز دهی بین key ها و query ها میتوانیم مقدار نزدیکی و هم جهت بودن key ها را با query های متناظر محاسبه کنیم. برای مثال اگر key و query هم جهت باشند باید ضرب آنها مقداری مثبت داشته باشد و همینطور حاصل ضرب بین آنها نسبت به بقیه جفت key و query ها

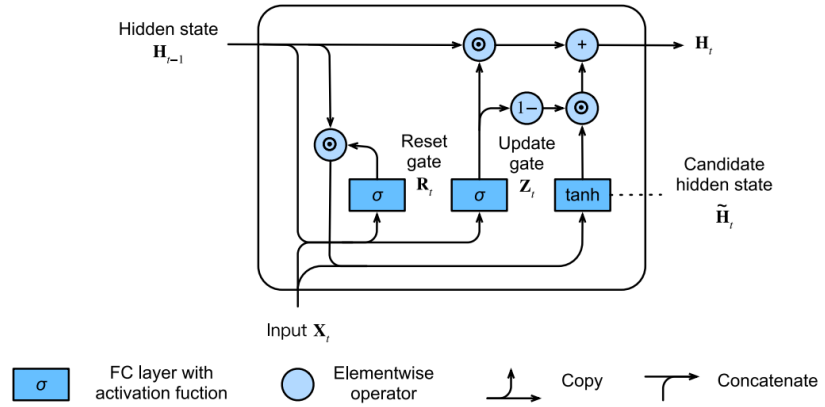
بیشتر است که با وارد شدن این مقدار در softmax اثر و وزن این مقدار حاصلضرب بیشتر هم میشود. پس به value متناظر با key که امتیاز بالاتری دارد وزن بیشتری هم داده میشود. جمع کردن این توانایی را برای سنجش و امتیاز دهی با استفاده از ضرب ندارد و صرف یک مقداز ثابت query را با key های مختلف جمع کرده و به تابع softmax وارد میکند پس در نهایت میزان وزن تخصیص یافته به هر key نمیتواند وزن مناسبی برای تناظر بین هر key با query باشد.

د)

فرض کنید که در GRU میخواهیم که خروجی هر مرحله فقط به یکی از ورودی‌های مراحل قبل بستگی داشته باشد. یعنی مثلا برای تولید th فقط از t'x به عنوان ورودی استفاده کرده باشیم که t'<t است. در این حالت مقادی ر مناسب برای گی تهای reset و update در هر مرحله چه باید باشد؟ پاسخ خود را به صورت دقیق توضیح دهید.

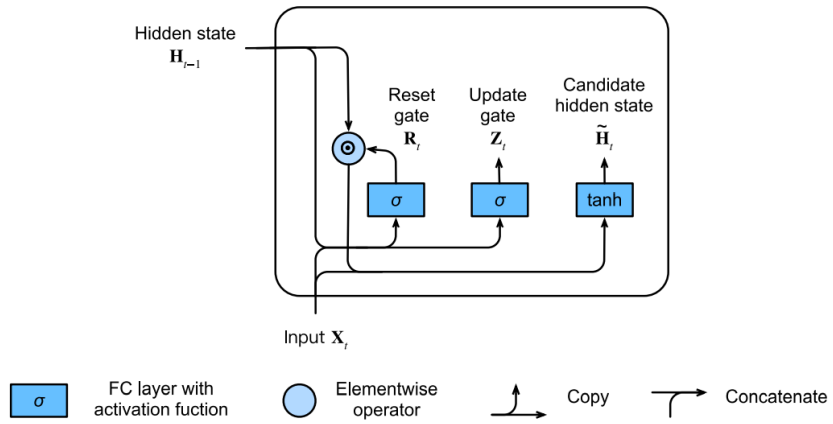
برای جواب این سوال از یک شکل در کتاب استفاده میکنیم.

$$\mathbf{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{H}}_t. \tag{9.1.4}$$



در شکل بالا ملاحضه میکنید که خروجی هر مرحله به مرحله قبلی وابسته است و نه مرحله فعلی. با توجه به فرمول update کردن gate با یک شدن مقدار zt ما صرفا state قدیمی را استفاده میکنیم. و اگر این مقداز صفر باشد state فعلی مد نظر ما خواهد بود. درمورد reset gate هم میتوان اظهار نظر کرد که چون قصد داریم hidden state فعلی را با قبلی جایگزین کنیم و فقط به یک ورودی قبلی وابسته هستیم باید مقدار Rt را صفر کنیم. چون اگر مقدار Rt صفر باشد ما صرفا با Xt کار میکنیم و اگر یک باشد ما با Xtدر کنار H_{t-1} نیز طرف هستیم.

$$\tilde{\mathbf{H}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xh} + (\mathbf{R}_t \odot \mathbf{H}_{t-1}) \mathbf{W}_{hh} + \mathbf{b}_h). \tag{9.1.3}$$



۵)

اگر در GRU فقط گیت update را پیاده‌سازی کنیم و گیت reset را حذف کنیم، چه اتفاقی می‌افتد و خروجی چه تغییراتی خواهد داشت؟

در مقاله مرجع [3] به این مسئله پرداخته است که اگر در مدل GRU، گیت reset را حذف کنیم چه اتفاقی خواهد افتاد. مدلی که در این مقاله معرفی میشود JANET نام دارد. مقایسه فرمول‌های روش GRU و JANET در جدول زیر قابل مشاهده است. همانطور که مشخص است با حذف عبارت r^t، GRU به مدل JANET تبدیل میشود.

GRU	JANET
$r^t = sigmoid(W_{rx}x^t + W_{rh}h^{t-1} + b_r)$ $z^t = sigmoid(W_{zx}x^t + W_{zh}h^{t-1} + b_z)$ $n^t = \tanh(W_{nx}x^t + W_{nh}(r^t \odot h_{t-1}) + b_n)$ $h^t = (1 - z^t) \odot n^t + z^t \odot h^{t-1}$	$\mathbf{f}_t = \sigma(\mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{W}_f\mathbf{x}_t + \mathbf{b}_f)$ $\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + (\mathbf{1} - \mathbf{f}_t) \odot \tanh(\mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{W}_c\mathbf{x}_t + \mathbf{b}_c)$ $\mathbf{h}_t = \mathbf{c}_t$

نتایج جدول زیر نشان میدهد که JANET عملکرد بهتری از LSTM بر روی کارهای حافظه مصنوعی در دیتاست های MNIST، pMNIST و MIT-BIH دارد.

Model	MNIST	pMNIST	MIT-BIH
JANET	**99.0** $\pm$ 0.120	**92.5** $\pm$ 0.767	**98.2** $\pm$ 0.109
LSTM	98.5 $\pm$ 0.183	91.0 $\pm$ 0.518	98.0 $\pm$ 0.070
RNN	10.8 $\pm$ 0.689	67.8 $\pm$ 20.18	77.4 $\pm$ 6.652
uRNN (Arjovsky et al., 2016)	95.1	91.4	-
iRNN (Le et al., 2015)	97.0	82.0	-
tLSTM* (He et al., 2017)	**99.2**	**94.6**	-
stanh RNN (Zhang et al., 2016)	98.1	94.0	-

* Effectively has more layers than the other networks.

در قسمت نتیجه گیری مقاله نیز ذکر شده است که حذف گیت reset از GRU نه تنها باعث کاهش عملکرد مدل نمیشود بلکه در بعضی اوقات باعث بهبود نتایج نیز میشود. حال به صورت دقیق تری به تغییرات خروجی مدل میپردازیم. به طور شهودی، استفاده از اطلاعات ورودی، کمی بیشتر از مقدار فراموش شده، تجزیه و تحلیل دنباله ورودی را آسان تر میکند. این موضوع شهودی در عمل نیز با کم کردن مقدار از پیش تعیین شده β از دنباله ورودی اثبات میشود.

$$
\begin{aligned}
\mathbf{s}_t &= \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{W}_f \mathbf{x}_t + \mathbf{b}_f \\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{W}_c \mathbf{x}_t + \mathbf{b}_c) \\
\mathbf{c}_t &= \sigma(\mathbf{s}_t) \odot \mathbf{c}_{t-1} + (\mathbf{1} - \sigma(\mathbf{s}_t - \beta)) \odot \tilde{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{c}_t .
\end{aligned}
$$

سوال ۴.

لطفا شبکه‌های زیر را پیاده‌سازی کنید و نتایج را با هم مقایسه و تحلیل کنید.

جدول نتایج مدل های مختلف در زیر آمده است:

Model	Accuracy		
	Single Layer	2 Layers	
		Without Dropout	With Dropout (0.2)
Simple RNN	0.6010	0.4462	0.5406
LSTM	0.6213	0.6340	**0.6359**
GRU	0.6213	0.6217	0.6159

مراجع

1) https://medium.com/heuritech/attention-mechanism-5aba9a2d4727

2) Yang, X. (2020, December). An overview of the attention mechanisms in computer vision. In *Journal of Physics: Conference Series* (Vol. 1693, No. 1, p. 012173). IOP Publishing.

3) Van Der Westhuizen, J., & Lasenby, J. (2018). The unreasonable effectiveness of the forget gate. *arXiv preprint arXiv:1804.04849*.

4) tutorialexample.com/can-we-remove-reset-gate-in-gru/

5) Tu, Z., Lu, Z., Liu, Y., Liu, X., & Li, H. (2016). Modeling coverage for neural machine translation. *arXiv preprint arXiv:1601.04811*.

6) Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

7) Levine, Y., Wies, N., Sharir, O., Bata, H., & Shashua, A. (2020). Limits to depth efficiencies of self-attention. *Advances in Neural Information Processing Systems*, *33*, 22640-22651.